APPENDIX C

```
nam spac  System.St rage
{

  public abstract class FindResult : IAsyncObjectReader
  {

    public FindResult();

    // Moves the FindResult to the next position in the result.
    public bool Read();
    public IAsyncResult BeginRead( AsyncCallback callback, object state );
    public bool EndRead( IAsyncResult asyncResult );

    // The current object.
    public object Current {get;}

    // Returns whether or not the FindResult contains any objects.
    public bool HasResults {get;}

    // Returns whether or not the FindResult is closed.
    public bool IsClosed {get;}

    // Returns the type of items in this FindResult.
    public Type ObjectType {get;}

    // Closes the FindResult
    public void Close();
    void IDisposable.Dispose();

    // Returns an enumerator over the FindResult, starting at the current position. Advancing
    // any enumerator on the FindResult advances all enumerators as well as the FindResult
    // itself.
    IEnumerator IEnumerable.GetEnumerator();
    public FindResultEnumerator GetEnumerator();

  }

  public abstract class FindResultEnumerator : IEnumerator, IDisposable
  {

    public abstract object Current { get; }
    public abstract bool MoveNext();
    public abstract void Reset();
    public abstract void Close();

    void IDisposable.Dispose();

  }

}

namespace System
```

```
{

 // A common interface for iterating over objects.
 public interface IObj ctR ad r : IEnumerable, IDisposabl
 {

   object Current {get;}
   bool IsClosed {get;}
   bool HasResults {get;}
   Type ObjectType {get;}

   bool Read();
   void Close();

 }

 // Adds asynchronous methods to IObjectReader
 public interface IAsyncObjectReader : IObjectReader
 {

   IAsyncResult BeginRead( AsyncCallback callback, object state );
   bool EndRead( IAsyncResult result );

 }
}
```

[Remainder of Page Intentionally Left Blank]